## 1) 准备好如下jar包:

Oracle数据库的jar:

Ojdbc14.jar

struts2的jar:

commons-pool.jar

freemarker-2.3.8.jar

ognl-2.6.11.jar

struts2-core-2.0.11.jar

xwork-2.0.4.jar

struts2整合spring的jar:

struts2-spring-plugin-2.0.9.jar

Hibernate整合需要的jar:

commons-logging-1.0.4.jar

## 2) Jndi连接池配置(以Tomcat为例)

<Resource name="jdbc/test" auth="Container"

type="javax.sql.DataSource"

maxActive="100" maxIdle="30" maxWait="10000"

username="EHT" password="eht"

driverClassName="oracle.jdbc.driver.OracleDriver"

url="jdbc:oracle:thin:@localhost:1521:Orcl"/>

## 3) 创建工程,加入准备好的jar包,

通过IDE自动导入Spring,Hibernate框架,

删除asm-2.2.3.jar这个冲突的Jar

创建

model,

dao,

service,

action,

inteceptor包,

建立相应的类:

model类,

dao类(继承HibernateDaoSupport),

service类,

Action类(继承ActionSupport),

interceptor类(依据拦截方式实现不同接口,常用的是实现MethodInteceptor环绕拦截接口);

## 4) 修改web.xml,加入Spring的装载器与Struts2.0的过滤器,编码过滤器:

<!-- Spring的装载器 -->

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

<!--ContextLoaderListener会去找contextConfigLocation这个参数的，选择spring的配置文件 -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:/applicationContext.xml</param-value>

</context-param>

<!-- struts的过滤器 -->

<filter>

<filter-name>struts2</filter-name>

<filter-class>

org.apache.struts2.dispatcher.FilterDispatcher

</filter-class>

</filter>

<filter-mapping>

<filter-name>struts2</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 著名的Character Encoding filter,设置转换编码为gbk -->

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>

org.springframework.web.filter.CharacterEncodingFilter

</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>gbk</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>\*.action</url-pattern>

</filter-mapping>

## 5) 创建struts.xml文档;

## 6) 在根目录下创建daoContext.xml,strutsContext.xml,serviceContext.xml,aopContext.xml;

## 7) 为上步创建的4个文档引入如下申明:

<?xml version="1.0" encoding="UTF-8"?>

<beans

xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

</beans>

## 8) 创建JSP界面,建立表单,定义好表单的action属性(与下一步的<action/>配置的name属性匹配);

若要使用struts2标签,可引入<%@ taglib prefix="s" uri="/struts-tags" %>

## 9) 配置struts.xml:

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"

"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

<!-- GBK编码 -->

<constant name="struts.i18n.encoding" value="GBK" />

<!--通过Spring管理Struts-->

<constant name="struts.objectFactory" value="spring" />

<package name="default" extends="struts-default">

<action name="user" class="loginAction">

<result name="input">/login.jsp</result>

<result name="error">/login.jsp</result>

<result name="success">/index.jsp</result>

<interceptor-ref name="params"></interceptor-ref>

</action>

</package>

</struts>

## 10) 配置strutsContext.xml:

<!—未加AOP的配置-->

<bean id="loginAction" class="com.company.action.LoginAction" scope="prototype">

<property name="userService" ref="userService" />

</bean>

## 11) 配置serviceContext.xml:

<!—未加事务的配置-->

<bean id="userService" class="com.company.service.UserService">

<property name="userDao" ref="userDao"/>

</bean>

## 12) 配置daoContext.xml:

<!-- 数据源配置 -->

<bean id="dataSource"

class="org.springframework.jndi.JndiObjectFactoryBean">

<property name="jndiName"

value="java:comp/env/jdbc/test" />

<property name="lookupOnStartup" value="false" />

<property name="cache" value="true" />

<property name="proxyInterface" value="javax.sql.DataSource" />

</bean>

<!-- sessionFactory -->

<bean id="sessionFactory"

class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

<property name="dataSource">

<ref bean="dataSource" />

</property>

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect">

org.hibernate.dialect.Oracle9Dialect

</prop>

<prop key="hibernate.show\_sql">true</prop>

</props>

</property>

<property name="mappingDirectoryLocations">

<list>

<value>classpath:/com/company /model/User.hbm.xml</value>

</list>

</property>

</bean>

<bean id="userDao" class="com.company.dao.UserDao">

<property name="sessionFactory" ref="sessionFactory"/>

</bean>

## 13) 修改自动生成的applicationContext.xml:

<?xml version="1.0" encoding="UTF-8"?>

<beans

xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

<import resource="aopContext.xml"/>

<import resource="daoContext.xml"/>

<import resource="serviceContext.xml"/>

<import resource="strutsContext.xml"/>

</beans>

## 14) src下加入Log4J资源文件: log4j.properties

## 15) 可以开始写业务逻辑与数据库操作方法了.

Action类中获得各范围对象的方法：

### 1.获得Session对象:

Map sessionMap = ActionContext.getContext().getSession();

//将信息放入session中

sessionMap.put("user", "valid");

### 2.获得request对象

HttpServletRequest request = ServletActionContext.getRequest();

request.setAttribute("username", "helloworld");

### 3.获得response对象

HttpServletResponse response = ServletActionContext.getResponse();

Cookie cookie = new Cookie("username",this.getUsername());

//如果设置为负数,则为会话Cookie;

cookie.setMaxAge(1000);

response.addCookie(cookie);

### 字段驱动:

本例采用字段驱动, 它一般用在页面表单比较简单的情况使用.界面表单文本框的name属性必须采用user.username形式

### 模型驱动:

界面表单文本框比较复杂, 用普通java对象充当模型部分

struts.xml的action配置加上<interceptor-ref name="model-driven"/>

并且让Action类实现ModelDriven接口,重写getModel()方法

public Object getModel(){

return user;

}

将JSP界面表单文本框的name属性的user.去掉

### 附加过滤器:

<!-- 容器负责session的开启和关闭 -->

<filter>

<filter-name>hibernateFilter</filter-name>

<filter-class>

org.springframework.orm.hibernate3.support.OpenSessionInViewFilter

</filter-class><!-- singleSession默认为true,若设为false则等于没用OpenSessionInView -->

<init-param>

<param-name>singleSession</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>hibernateFilter</filter-name>

<url-pattern>\*.action</url-pattern>

</filter-mapping>